CERTIFICATE OF TRANSMISSION

I hereby certify that this correspondence is being transmitted by either submission using the EFS WEB submission system, fax to the U.S. Patent and Trademark office to fax number 571-273-8300, or is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 to on March 27, 2008.

/Brian C. Kunzler/
Attorney for Appellant

Docket No.: TUC920030061US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant:	YU-CHENG HSU, et al.	
Serial No.:	10/641,377	
Filed:	08/14/2003	Group Art Unit: 2192
For:	INTEGRATED SOURCE CODE DEBUGGING APPARATUS METHOD AND SYSTEM	
Examiner:	WEI, ZHENG.	

APPEAL BRIEF

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Examiner:

The USPTO received Appellant's Notice of Appeal on 28 January 2008, which was filed in response to an Advisory Action mailed 27 December 2007. Appellant appeals the rejection of pending claims 1-26.

This Brief is being filed under the provisions of 37 C.F.R. § 41.37. The filing fee set forth in 37 C.F.R. § 41.20(b)(2) of $510.00 is submitted herewith. The Commissioner is hereby authorized to charge payment of any additional fees associated with this communication, or to credit any overpayment, to Deposit Account No. 090449.

1. REAL PARTY IN INTEREST

The real party in interest is the assignee, International Business Machines Corporation, Armonk, New York.

2. RELATED APPEALS AND INTERFERENCES

There are no related appeals, interferences, or judicial proceedings.

3. STATUS OF CLAIMS

The Office Action mailed on 01 June 2007 and the Advisory Action mailed 27 December 2007 maintained the rejection of claims 1-7, 9-12, 15-17, 19 and 20 under 35 U.S.C. 102(b) as being anticipated by Testardi (6,249,882), and claims 8, 13, 18, and 25 under 35 USC 103(a) as being unpatentable over Testardi in view of Rosenburg (Jonathan B. Rosenberg, "How debuggers work"), and claims 14 and 26 as being unpatentable over Testardi in view of Frascone (David Frascone, "Debugging kernel modules with user-mode Linux"). Appellants appeal the rejection of all the claims but will focus arguments on independent claims 1, 9, 15, 19, and 21.

4. STATUS OF AMENDMENTS

All amendments have been entered the Examiner. The claims submitted on 01 August 2007 contained no amendments but were nevertheless entered by the examiner on 27 December 2007.

5. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed subject matter deals with a specific apparatus, system, method, and computer readable medium to initialize a target environment to a particular state suitable for debugging a target function. For example, the abstract of Appellant's Published Application No. US 2005/0039169 (hereinafter "Specification") describes the apparatus as follows:

> *An apparatus for debugging source code includes a source code debugger configured to display state information and one or more initialization routines corresponding to a particular software function. The initialization routines initialize*

a target environment to a particular system state and facilitate replication, isolation, and analysis of software coding errors. In one embodiment, a function selector facilitates selection of the target function by a user and generates an execution request. In turn, a task dispatcher dispatches the initialization routines and associated software function in response to the execution request. The present invention greatly simplifies interactive debugging of source code. Rather than generating complex, error-prone, and often timing-dependent manipulation sequences of registers, memory, peripheral devices, and the like, a user simply selects the initialization routines that generate the particular states and conditions necessary to replicate and analyze a particular software error.

Paragraphs 13 and 18 of the specification further elaborate that user is able to select which initialization routines are executed with a selected target function:

In one embodiment, the function selector is compiled into the application source code and is displayed on the target platform. In another embodiment, the function selector is integrated into the source code debugger (either on the target platform or on a host,) which sends a message to the task dispatcher to initiate execution of the selected initialization routines and target function within the target environment.

... In one embodiment, the user selects the initialization routines and associated target function in order to debug the target function in a particular target environment and associated states and conditions.

Paragraph 36 and 41 explain how the initialization routines provide functionality not available in traditional source code debuggers:

The initialization routines 220 may be custom developed by a developer to generate a specific state corresponding to an anticipated or discovered condition. Custom development facilitates changing state information not readily accessible via the debug interface of a source code debugger. In addition to anticipated or

discovered conditions, the initialization routines 220 may include state information collected from actual deployed systems such as deployed systems in which an error was detected.

...

...

The depicted source code debugging module 210 leverages the programming power available within a source code development system to the challenges of testing and debugging software code and provides power and flexibility currently not found in integrated debugging environments.

Appellants submit that the claimed invention provides functionality not found in the prior art of Testardi and others because the prior art does not include all of the elements of Appellants claimed invention. The following quotation of independent claims 1, 9, 15, 19, and 21 cite those elements and include reference numerals and parenthetical references to representative examples of the elements and components in compliance with 37 CFR 41.37(c)(1)(v).

1. An apparatus for debugging source code, the apparatus comprising:

a source code debugger (*110 on Fig. 2*) configured to display state information; [*see, for example, paragraphs 30, 31, 34, and 45*] and

at least one initialization routine (*220 on Figs 2 and 3*) configured to initialize a target environment (*120 on Fig. 2*) to a particular state [*see, for example, paragraphs 35-37*], the at least one initialization routine selectively coupled (*via Function Selector 310 on Fig. 3*) to a target function (*160 on Figs 2 and 3*) within a target application (*150 on Fig. 2*). [*see, for example, paragraphs 13 and 43*]

9. A method for debugging source code, the method comprising:

dispatching (*420 on Fig. 4 and 320 on Fig. 3*) at least one initialization routine (*220 on Figs 2 and 3*) selectively coupled to a target function (*160 on Figs 2 and 3*), the at least one initialization routine configured to initialize a target environment to a particular state; [*see, for example, paragraphs 13, 35-37, and 43*]

dispatching (*430 on Fig. 4 and 320 on Fig. 3*) the target function; [*see, for example, paragraphs 44 and 39-40*] and

displaying (*440 on Fig. 4 and 230 on Fig. 2*) state information within a source code debugger. [*see, for example, paragraph 45*]

15. An apparatus for debugging source code, the apparatus comprising:

means for dispatching (*320 on Fig. 3 and 420 on Fig. 4*) at least one initialization routine (*220 on Figs 2 and 3*) selectively coupled to a target function (*160 on Figs 2 and 3*), the at least one initialization routine configured to initialize a target environment (*120 on Fig. 2*) to a particular state; [*see, for example, paragraphs 13, 35-37, and 43*]

means for dispatching (*320 on Fig. 3 and 430 on Fig. 4*) the target function; [*see, for example, paragraphs 39-40 and 44*] and

means for displaying (*230 on Fig. 2, 130 on Fig. 1, and 440 on Fig. 4*) state information.

19. A system for debugging source code, the system comprising:

a target environment (*120 on Fig. 2*) comprising a target platform (*140 on Fig. 2*) including an operating system and a target application (*150 on Fig. 2*);

a source code debugger (*110 on Fig. 2*) configured to display state information; [*see, for example, paragraphs 30, 31, 34, and 45*] and

at least one initialization routine (*220 on Figs 2 and 3*) configured to initialize the target environment to a particular state [*see, for example, paragraphs 35-37*], the at least one initialization routine selectively coupled to a target function (*160 on Figs 2 and 3*) within the target application. [*see, for example, paragraphs 13 and 43*]

21. A computer readable storage medium comprising computer readable program code for debugging source code, the program code configured to:

enable selection (*via Function Selector 310 on Fig. 3*) of a target function (*160 on Figs 2 and 3*); [*see, for example, paragraphs 13 and 43*]

dispatch at least one initialization routine (*220 on Figs 2 and 3*) selectively coupled to the target function, the at least one initialization routine configured to initialize a target environment (*120 on Fig. 2*) to a particular state; [*see, for example, paragraphs 13, 35-37, and 43*] and

dispatch the target function. [*see, for example, paragraphs 39-40 and 44*]

In summary, the claimed invention enables a user to selectively couple initialization routines to a target function and dispatch the initialization routines and thereby initialize a target application and/or environment to a particular state suitable for debugging the target function.

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

I. Whether the Examiner has established a *prima facie* case of anticipation of claims 1, 9, 15, 19, and 21 under 35 U.S.C. 102(b) as being anticipated by Testardi (6,249,882).

7. ARGUMENT

II. The Examiner has not established a *prima facie* case of anticipation of claims 1, 9, 15, 19, and 21 under 35 U.S.C. 102(b) as being anticipated by Testardi (U.S. Patent No. 6,249,882).

Appellants' arguments will focus on the elements of independent claim 1 with the understanding that the same arguments apply to the elements delineated in independent claims 9, 15, 19, and 21.

A review of the present invention may help clarify the novelty of the Appellants' claims over the cited prior art. As shown in Figures 2-4 an apparatus for debugging source code enables a user to invoke selected initialization routines corresponding to a particular function within a source code debugger in order to initialize a target environment to a state suitable for executing the function and observing its performance with a debugger.

Much of the utility of the present invention arises from the relationship between the target function and the initialization routines. The correspondence between the initialization routines and the target functions may be dynamically adjusted to according to user preference and need (*see paragraph 43*). More than one set of initialization routines may be associated with a target function. The initialization routines may include function independent routines (suitable for use with any target function) as well function dependent routines (suitable for use with specific target functions). The initialization routines may be developed using the same programming languages and tools as the application source code used to generate the target functions that are being tested. Furthermore, compiled initialization routines and compiled target functions may co-exist as compiled binary files within the system under test and selectively paired and executed (*see Figures 2 and 3 and the associated description*). Effectively, the present invention enables development or test personnel to create an arsenal of initialization routines and then selectively apply the initializations routines to a particular function in order to achieve a desired effect.

In contrast to the present invention, the cited prior art extracts test directives and parameters that are embedded as comments within the source code files of an application (*see the Abstract of*

Testardi as well as Figure 1 and the associated description in col. 5 line 48 to col. 6 line 44). The test directives are interpreted by an interpreter 110 in order to initialize the environment and construct a command script that is executed by the debugger 112 (*see also col. 10 lines 4-52*).

The Examiner uses column 9 lines 13-15 of Testardi to reject Appellants claims along with Fig. 2 item 206 and Fig. 3 item 304. Appellants welcome the opportunity to review these particular citations in detail. Column 9 lines 13-15 discloses "... the debugger tool is used to invoke particular functions within a program and to set particular variables within the program for testing purposes." The description of item 206 indicates that "The test directive and parameters so extracted by operation of element 204 are then interpreted by element 206 for purpose of setting up a required environment of the computing system for operation of the test." The description of item 304 indicates that "Element 304 is then operable to set up any required system environment parameters in accordance with parameters defined by the test sequence as extracted in element 300."

While Testardi may be used to achieve some of the same results as the claimed invention, Appellants assert that the Testardi does not disclose all of the limitations of Appellants claims. Appellants assert that Testardi embeds test directives and parameters within source code and provides an interpreter to execute those test directives such as element 206 or 110. Therefore the debugging of Testardi is simply an interactive debugging environment with an interpreter that executes test directives embedded in the source code.

Appellants assert that an interpreter of embedded test directives such as disclosed in Testardi suffers from many of the shortcomings disclosed in background paragraphs 6-8 and does not read on the present claims. For example, claim 1 cites "at least one initialization routine configured to initialize a target environment to a particular state, the at least one initialization routine selectively coupled to a target function within a target application". Appellants assert that an initialization routine provides additional functionality not provided by interpreted test directives.

For example, the initialization routine(s) may be custom developed by a developer using the same tools as the application (see paragraphs 36 and 41) and thereby provide all of the API interfaces, library routines, compiled performance, and the like that are available to applications. In contrast, the use of test directives and an interpreter restricts the user to the specific commands supported by the interpreter and limits the attainable performance and timing control of the

debugging system. For example, Appellants note that the test directives shown in the 'C' source file example shown in column 22 (i.e. lines 56-61) of Testardi are embedded as comments within the source code. Furthermore, the embedded test directives do not conform to the syntax of a routine written in 'C' and therefore could not be compiled into an initialization routine by the compiler. Therefore the test directives are unable to harness the functionality of the available source code programming environment and are limited to the functionality provided by an interpreter.

Given the foregoing, Appellants assert that "at least one initialization routine configured to initialize a target environment to a particular state" does not read on an interpreter that responds to test directives embedded within source code as disclosed by Testardi and the rejection under 35 U.S.C. 102(b) is improper.

SUMMARY

In view of the foregoing, Appellants respectfully assert that each of the claims on appeal have been improperly rejected because the Examiner has failed to show that Testardi anticipates each of the limitations of independent claims 1, 9, 15, 19, and 21. In particular, Testardi does not disclose "at least one initialization routine configured to initialize a target environment to a particular state, the at least one initialization routine selectively coupled to a target function within a target application".

Respectfully submitted,

Date: March 27, 2008

/Brian C. Kunzler/

Kunzler & Associates
8 E. Broadway, Suite 600
Salt Lake City, Utah 84101
Telephone: 801/994-4646

Brian C. Kunzler
Reg. No. 38,527
Attorney for Appellant

8. CLAIMS APPENDIX

1. An apparatus for debugging source code, the apparatus comprising:

a source code debugger configured to display state information; and

at least one initialization routine configured to initialize a target environment to a particular state, the at least one initialization routine selectively coupled to a target function within a target application.

2. The apparatus of claim 1, further comprising a task dispatcher configured to dispatch the at least one initialization routine in response to an execution request.

3. The apparatus of claim 1, further comprising a function selector configured to generate an execution request in response to selection of the target function by a user.

4. The apparatus of claim 3, wherein the function selector is integrated into the source code debugger.

5. The apparatus of claim 1, wherein the particular state corresponds to an application error.

6. The apparatus of claim 1, further comprising a deployed system configured to dump information used to initialize the target environment to the particular state.

7. The apparatus of claim 1, wherein the at least one initialization routine comprises a function independent initialization routine and a function dependent initialization routine.

8. The apparatus of claim 1, wherein the source code debugger is further configured to single step through the target function.

9. A method for debugging source code, the method comprising:

dispatching at least one initialization routine selectively coupled to a target function, the at least one initialization routine configured to initialize a target environment to a particular state;

dispatching the target function; and

displaying state information within a source code debugger.

10. The method of claim 9, further comprising collecting state information from a deployed environment.

11. The method of claim 9, further comprising collecting state information in response to an application error.

12. The method of claim 9, wherein dispatching the at least one initialization routine comprises dispatching a function independent initialization routine and a function dependent initialization routine.

13. The method of claim 9, further comprising single stepping through the target function.

14. The method of claim 9, further comprising recompiling kernel mode code into user mode code.

15. An apparatus for debugging source code, the apparatus comprising:

means for dispatching at least one initialization routine selectively coupled to a target function, the at least one initialization routine configured to initialize a target environment to a particular state;

means for dispatching the target function; and

means for displaying state information.

16. The apparatus of claim 15, further comprising means for collecting state information from a deployed environment.

17. The apparatus of claim 15, further comprising means for collecting state information in response to an application error.

18. The apparatus of claim 15, further comprising means for single stepping through the target function.

19. A system for debugging source code, the system comprising:

a target environment comprising a target platform including an operating system and a target application;

a source code debugger configured to display state information; and

at least one initialization routine configured to initialize the target environment to a particular state, the at least one initialization routine selectively coupled to a target function within the target application.

20. The system of claim 19, further comprising a deployed system configured to provide information used to initialize the target environment to the particular state.

21. A computer readable storage medium comprising computer readable program code for debugging source code, the program code configured to:

enable selection of a target function;

dispatch at least one initialization routine selectively coupled to the target function, the at least one initialization routine configured to initialize a target environment to a particular state; and

dispatch the target function.

22. The computer readable storage medium of claim 21, wherein the method further comprises collecting state information from a deployed environment.

23. The computer readable storage medium of claim 21, wherein the method further comprises collecting state information in response to an application error.

24. The computer readable storage medium of claim 21, wherein dispatching the at least one initialization routine comprises dispatching a function independent initialization routine and a function dependent initialization routine.

25. The computer readable storage medium of claim 21, wherein the method further comprises single stepping through the target function.

26. The computer readable storage medium of claim 21, wherein the method further comprises recompiling kernel mode code into user mode code.